CHAPTER 2

Anatomy of the Internet of Things

It may appear to be a daunting task to engineer a new networking architecture for the Internet of Things (IoT). Yet nothing less than a completely new approach is needed. The Internet of Things environment is so different, and the devices to be connected so varied, that there has never been a networking challenge quite like it since the origin of what is now called the Internet.

In developing this new architecture for the Internet of Things, key lessons have been drawn from the development of the traditional Internet and other transformational technologies to provide some basic guiding principles:

- It should specify as little as possible and leave much open for others to innovate.
- Systems must be designed to fail gracefully: seeking not to eliminate errors, but to accommodate them.
- Graduated degrees of networking functionality and complexity are applied only where and when needed.
- The architecture is created from simple concepts that build into complex systems using the analog provided by natural phenomena.
- Meaning may be extracted from data in real time.

The emerging architecture for the Internet of Things is intended to be more inclusive of a wider variety of market participants by reducing the amount of networking knowledge and resources needed at the edges of the network. This architecture must also be extremely tolerant of failures, errors, and intermittent connections at this level. (Counter intuitively, the best approach is to *simplify* protocols at the edge rather than to make them more complex.)

In turn, increasing sophistication of networking capabilities are applied at gateways into the traditional Internet, in which propagator nodes provide communications services for armies of relatively unsophisticated devices.

Finally, meaning can be extracted from the universe of data in integrator functions that provide the human interface to the Internet of Things. This level of oversight is applied only at the highest level of the network; simpler devices, like worker bees in a hive, need not be burdened with computational or networking resources.

To explore what's needed for this new architecture, it is first necessary to *abandon* the networking status quo.

Traditional Internet Protocols Aren't the Solution for Much of the IoT

When contemplating how the Internet of Things will work, it helps to *forget* the conventional wisdom regarding traditional networking schemes—especially wide area networking (WAN) and wireless networking. In traditional WAN and wireless networking, the bandwidth or spectrum is expensive and limited, and the amount of data to be transmitted is large and always growing. Although over-provisioning data paths in wiring the desktop (and a majority of the traditional Internet) is commonplace, this isn't usually practical in the WAN or wireless network—it's just too expensive. With carriers largely bearing the cost and passing it along to customers, wireless costs range as high as ten times the wired equivalents using IP.

Besides cost, there's the matter of potential data loss and (in the wireless world) collisions. Traditional networking protocols include lots of checks and double-checks on message integrity to minimize costly retransmissions. These constraints led to today's familiar protocol stacks, such as TCP/IP and 802.11.

Introducing the "Chirp"

In most of the Internet of Things, however, the situation is completely different. The costs of wireless and wide-area bandwidth are still high, to be sure. And because many of the connections at the edge of the network—the IoT frontier, so to speak—will be wireless and/or lossy, any Internet of Things architecture must address these factors. But the amounts of data from most devices will be almost *immeasurably low* and the delivery of any single message *completely uncritical*. As discussed previously, the IoT is lossy and intermittent, so the end devices will be designed to function perfectly well even if they miss sending or receiving data for a while—even for a long while. As discussed earlier, it is this self-sufficiency that eliminates the criticality of any *single* message.

After reviewing all existing options in considering the needs of the IoT architecture from the ground up, it is clearly necessary to define a new type of data frame or packet. This new type of packet offers only the amount of overhead and functionality needed for simple IoT devices at the edge of the network—*and no more*. These small data packets, which are called *chirps,* are the fundamental building block of the emerging architecture for the IoT. Chirps are different from traditional Internet protocol packets in many ways (see the "Why Not the IP for the IoT?" sidebar. Fundamental characteristics of chirps include the following:

- Chirps incorporate only minimal overhead payloads, "arrows" of transmission (see below), simple *non-unique* addresses, and modest checksums.
- Chirps are inherently individually noncritical by design.
- Therefore, chirps include no retransmission or acknowledgment protocols.

Any additional functions necessary for carrying chirp traffic over the traditional Internet, such as global addressing, routing, and so on, are handled autonomously by other network devices by means of adding information to received simple chirps. There are therefore no provisions made for these functions within a chirp packet.

Lightweight and Disposable

In contrast to traditional networking packet structures, IoT chirps are like pollen or bird songs: lightweight, broadly propagated, and with meaning only to the "interested" integrator functions or end devices. The IoT is receiver-centric, not sender-centric, as is IP. Because IoT chirps are so small and *no individual chirp is critical*, there is limited concern over retries and resulting broadcast storms, which are a danger in IP.

It's true that efficient IoT propagator nodes will prune and bundle broadcasts (see Figure 2-1 and Chapter 4), but seasonal or episodic broadcast storms from end devices are much less of a problem because the chirps are small (and thus cause less congestion) and individually uncritical. Excessive chirps may thus be discarded by propagator nodes as necessary.

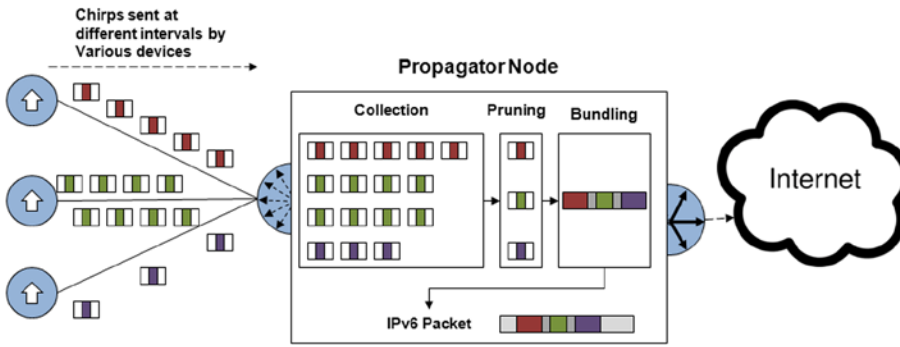

Figure 2-1. *Chirps are typically collected within propagator nodes, bundled and pruned as necessary for transmission, and then typically forwarded via IPv6 over the traditional Internet*

Functionality the IoT Needs—and Doesn't

This very different view of networking means that huge packets, security at the publisher, and assured delivery of any *single* message are unnecessary, allowing for massive networks based on extremely lightweight components. In one sense, this makes the IoT more "female" (receiver-oriented) than the "male" structure of IP (sender-oriented).

But there is obviously no point in having an IoT if nothing *ever* gets through. How can the acknowledged unpredictable nature of connections be managed? The answer, perhaps surprisingly, is over-provisioning—but only very locally between chirp device and propagator node. That is, these short, simple chirps may be re-sent over and over again as a brute-force means of ensuring that some get through.

Efficiency Out of Redundancy

As seen in Figure 2-2, because the chunks of data are so small, the costs of this over-provisioning at the very edge of the IoT are infinitesimal. (They are often handled by local Wi-Fi, Bluetooth, infrared, and so on, so they are not metered by any carrier.) Therefore, the benefits of this sort of scheme are huge. Because no individual message is critical, there's no need for any error-recovery or integrity-checking overhead (except for the most basic checksum to avoid a garbled message). Each chirp message simply has an address, a short data field, and a checksum. In some ways, these messages are what IP datagrams were meant to be. Chirps are also similar in many ways to the concepts of the Simple Network Management Protocol (SNMP), with simple "get" and "set" functionality.

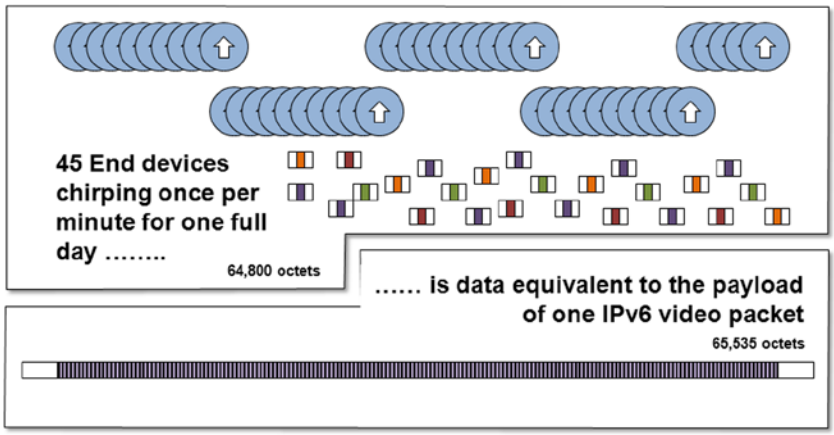

Figure 2-2. *Many small chirps (machine-to-machine–oriented) are still considerably less data than a much longer IP packet (human-oriented)*

Importantly, the cost and complexity burden on the end devices to incorporate chirp messaging will be very low–because it must be in the IoT. The most efficient integration schemes will likely be "chirp on a chip" approaches, with minimal data input/output and transmission/reception functionality combined in a simple standardized package.

The chirp will also incorporate the "arrow" of transmission mentioned previously, identifying the general direction of the message: whether toward end devices or toward integrator functions (see Figure 2-3). Messages moving to or from end devices need only the address of the end device; where it is headed or where it is from is *unimportant* to the vast majority of simple end devices. These devices are merely broadcasting and/or listening, and local relevancy or irrelevancy is all that matters.

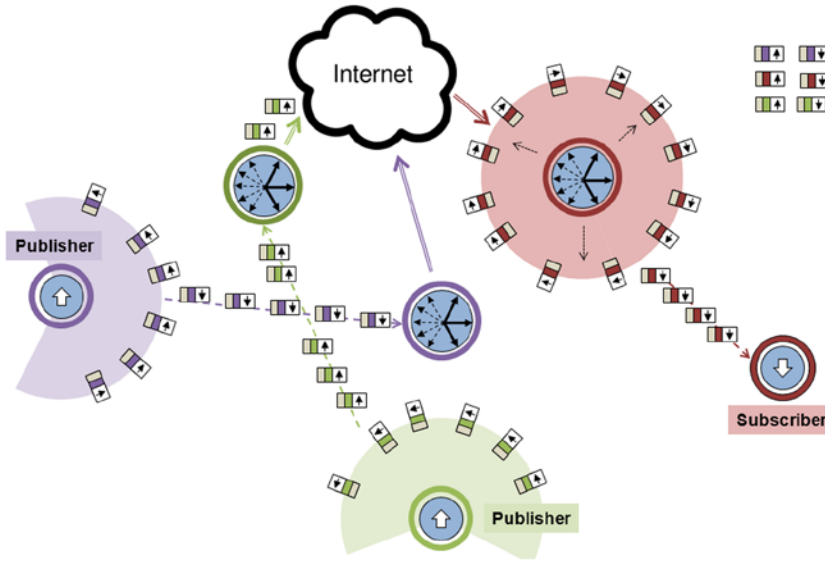

Figure 2-3. *Each chirp includes an "arrow" of transmission that indicates its direction of propagation: toward end devices or toward integrator functions*

So the end devices may be awash in the ebb and flow of countless transmissions. They may broadcast continuously and trust that propagator nodes and integrator functions elsewhere in the network will delete or ignore redundant messages. Likewise, they may receive countless identical messages before detecting one that has changed and requires an action in response.

In essence, this means that the chirp protocol is "wasteful" in terms of retransmissions only very locally, where bandwidth is cheap or free (essentially "off the net"). But because propagator nodes are designed to minimize the amount of superfluous or repeated traffic that is forwarded, WAN costs and traffic to the traditional Internet are vastly reduced.

Note that, unlike traditional network end devices such as smartphones and laptops, the largest percentage of IoT end devices likely *will not* include both send and receive functions (see Figure 2-4). An air quality sensor, for example, needs to send only the current state for whatever chemicals it is measuring. It begins sending when powered on, and repeatedly chirps this information until switched off. This may simplify significantly the hardware and embedded software needed at the vast majority of end points.

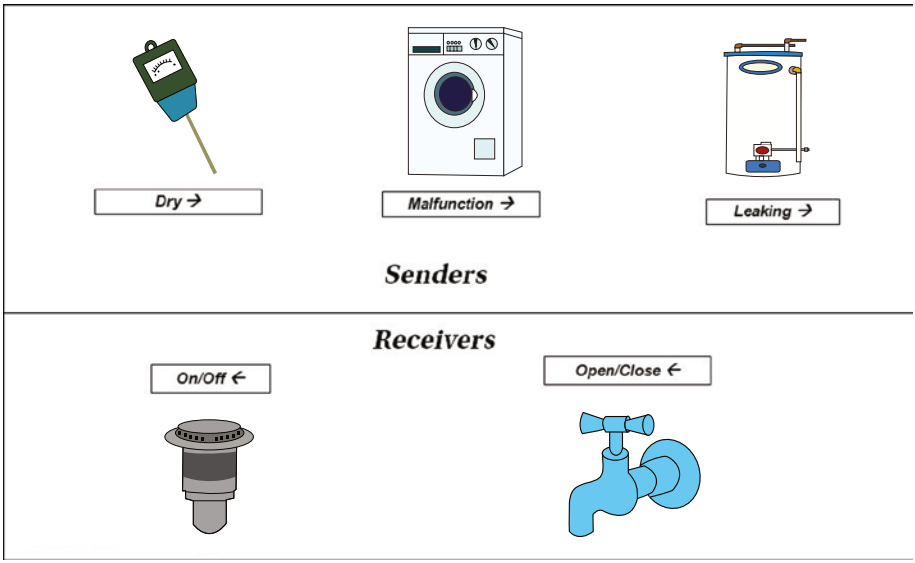

Figure 2-4. *Many IoT devices will be send-only or receive-only*

WHY NOT IP FOR THE IOT?

Although IPv6 already exists (and will at some point be ubiquitous within the traditional Internet), it is not the ideal format for much of the IoT traffic—for a variety of reasons outlined in Chapter 1 related to processing power and device memory that would be required in the tremendous quantity of otherwise simple and cheap end devices in the Internet of Things. But there are also fundamental protocol inefficiencies that make IPv6 unsuitable for the IoT, as discussed here. Still, there will be a vast array of end devices that must use IP, so a dual approach to protocols, IP, and the chirp protocols used together to service IoT devices of all kinds would yield an optimal result. It is worthwhile to compare and contrast the traditional IPv6 packet format with the IoT chirp, considering the difference in applications for which each is designed.

IP protocols were originally designed (in the early 1970s) for peer-to-peer communications between large hosts. These exchanges tended to be in large blocks of data, so IP is fundamentally oriented toward larger payloads. In addition, because WAN connections were extremely expensive and unreliable at the time when these host-to-host links were first designed, it was critical to incorporate the addresses of sender and receiver, as well as error detection and retransmission capabilities within the protocol to make it more robust. The result is that the header overhead of a single IPv6 packet is fairly high: 40 bytes. (A significant amount of the overhead in IP is dedicated to security, encryption, and other services, none of which matters at the very edges of the Internet of Things where the simplest devices predominate.)

Although originally imagined for machine-to-machine traffic, much of the IP traffic on the traditional Internet today is oriented toward human communications. This often consists of relatively long-duration sessions and some degree of full-duplex interaction over relatively costly links (at least until recently). Traditional networking protocols are thus designed for reliability and recoverability because nearly every packet is necessary for human context and understanding.

As a general-purpose protocol designed to carry data of virtually any type or degree of criticality, IP imposes at least this much overhead on every transmission. The structure of the header is strictly defined, and most aspects are unchangeable—the standard is absolute.

IP establishes Maximum Transmission Units (MTUs) that describe the maximum size of data blocks that the link is expected to carry. They have increased over time to 1,280 bytes for IPv6, although most deployed networks have MTUs of 1,500 or more. Peer-to-peer host traffic will tend to be managed by the applications to come in larger blocks to match outbound blocks to the MTU to maximize efficiency.

With packets of these sizes, the IP overhead is a relatively small percentage of the overall "cost" of transmission. For example, 40 bytes of IPv6 overhead added to a 1280 byte MTU is roughly 97% efficiency. In actual practice, the overhead is often doubled because an acknowledgment packet is required to be sent for each arriving packet. With no data payload, this acknowledgment packet is also the IPv6 minimum of 40 bytes. (In the host-to-host environment for which IP was originally designed, there would usually be some data to be sent in the return direction, though, so the overhead is not always wasted.)

But the Internet of Things is definitely not made up of peer-to-peer communications between like hosts. Because Internet of Things chirp traffic is machine-to-machine-oriented, it is by contrast sporadic, (nearly always) simplex, and almost free because of low volumes of data and low duty cycles. The IoT is a publish/subscribe model with very simple end devices transmitting or receiving only tiny amounts of individually noncritical pieces of data at one time. A temperature sensor output might be expressed in 8 bits or fewer, for example. So for a large number of similar applications, the data "payload" would be only 1 byte. Applying IPv6 to this application with the same overhead calculation yields 40 bytes of IPv6 overhead to 1 byte sensor data is only about 2% efficiency!

Chirps are designed to minimize overhead for this type of data in the multiple ways described in this chapter, such as simplifying addresses, eliminating retransmission overhead, and so on. Most importantly, the relative structure of the chip packet adds differing amounts of overhead depending on the type and size of the data generated by the end device, ensuring maximum efficiency. Only the smallest (4.5 byte total, 3.5 byte overhead) chirp packet would be needed to send an 8-bit payload, for an efficiency gain of roughly an order of magnitude over IPv6 (18% vs. 2%). See the comparison in Figure 2-5.

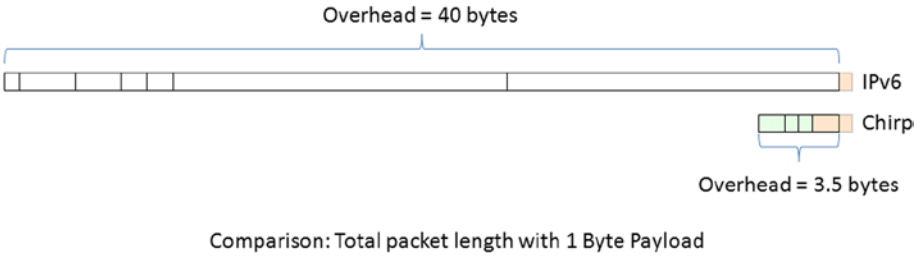

Figure 2-5. *Comparison of TCP/IP packet and chirp packet overheads for a 1-byte payload from a simple sensor*

In general, larger data payloads result in more efficient chirp packets, with the headers increasing only incrementally to match specific applications, as further described in Chapter 6. For example, a 4-byte end-device payload could be handled with the same 3.5-byte overhead, for an efficiency of more than 50%.

One other critical differences between chirps and IP packets is that chirps are self-classified through external markers (see "Family Types" below). This makes it easy for integrator functions to discover new interesting data flows by looking for affinities with "known" data sources. The only way this could be accomplished in IP would be to include the classification information within the payload, which would require impractical deep inspection of every packet by propagator nodes and integrator functions.

So chirps make eminent sense in the "last mile" of network connections at the edge of the IoT frontier instead of IPv6. Beyond the edges of the network, the situation changes, however. Propagator-to-propagator or propagator-to-integrator communications can much more resemble host-to-host traffic because their transmissions may consist of bundled chirps to and from many end devices (increasing the size of the data blocks to be exchanged). In those situations, the error correction and other features of a protocol such as IP are more useful, as more fully described in Chapter 4. And because these communications often use the traditional Internet as the medium, it makes even more sense to simply use existing IPv6 networking protocol stacks.

Note that some sensitive and proprietary applications (government, security, financial, and so on) will remain that also require the additional features of IP in terms of guaranteed delivery, security, and so on. These types of applications are not part of the emerging Internet of Things as defined in this book and will, of course, remain on traditional protocols.

It's All Relative

The detailed structure of the chirp packet is described in Chapter 6, but a brief introduction is useful here. The key difference between the Internet of Things packet and other packet formats is that the meanings of values within the packet are `_relative`. That is, there is no fixed definition for the packet locating headers, addresses, and so on (as there is for IPv6, for example).

As seen in Figure 2-6, *_markers* are used in place of a fixed format definition to allow receiving devices to determine information such as sending address, type of sensor and data, arrow of transmission, and so on. These markers are both `_public_` and `_private_` types.

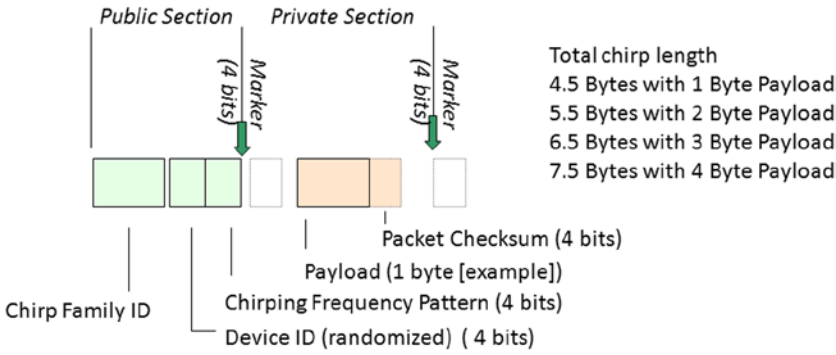

Figure 2-6. *The IoT chirp packet is unique in that addressing and other information is determined by relative position to defined markers, not by a rigid general overall protocol formats*

Public markers, which are found in every IoT packet, allow the receiving device to "parse" the incoming traffic. When a public marker is noted, the receiving device examines data ahead of and behind the marker for specific bits needed to determine how the rest of the packet will be forwarded and/or acted upon. The receiving device need not examine the packet except for the areas indicated by the location and type of public marker observed. Public markers include the basic arrow of transmission described previously, a limited 4-bit checksum for packet verification, and so on. Bits in the data field that are not part of the routing and verification information are simply treated as a data payload at this level of examination.

Format Flexibility

The presence of public markers within the IoT chirp packet permits the length of the IoT packet to vary as necessary for the specific application, device type, or message format. Different families of IoT packets with varying amounts of public data fields are defined to allow sufficient information to be added for applications that need additional context, but also to allow for minimal overhead for the most basic device types and generic IoT packet propagation.

The use of public markers is inspired by nature, including the transcription or "reading" of heredity information coded in DNA within genes to create proteins needed for development and life. DNA strands may contain repetitions and "junk" sections that should not be read, but localized markers are used to indicate "start" and "stop" points for transcription. Receiving devices use public markers in the same way to examine IoT chirp packets without requiring specific byte counts or other overhead-generating restrictions.

Private Markers for Customization and Extensibility

Private markers are permitted within the generic "data" field defined by public markers to allow customization of data formats for specific applications, manufacturers, and so on. As with public markers, the private marker allows a receiving device to parse the data stream to locate information for specific needs.

Addressing and "Rhythms"

As noted earlier, billions of end devices of the IoT will be extremely inexpensive and may be manufactured by makers throughout the world, many of whom will not have extensive networking knowledge. For this reason, ensuring address uniqueness through a centralized database of device addresses for the hundreds of billions of IoT end points is a nonstarter.

Part of the public information in the IoT chirp packet will be a simple, non-unique, 4-bit device ID applied through PC board traces, hardware straps, DIP switches, or similar means. As described in Chapter 6, it will combine with a randomly generated 4-bit pattern to ensure a much lower potential for two end devices, connected to the same local propagator node, to have identical identifications. (This combination of bits is also used to vary transmission rates in wireless environments to avoid a "deadly embrace.")

If additional addressing specificity and/or security is required in particular applications, it will be possible to add this information within the private space of the IoT packet.

Family Types

The final public information contained in all IoT chirp packets is a classification into one of 255 possible chirp "families." As described in Chapter 6, these families will primarily divide along type and application lines, such as sensors of various types, control valves, green/yellow/red status indicators, and so on. These chirp families will be defined from generic to more specific, and will be broad and extensible enough to allow any type of IoT application. As noted previously, for specific applications or devices in which more granularity of type classification is desired, this custom information may be defined by private markers within the data field.

The type and classification of the chirp packets enables one of the most far-reaching benefits of the IoT: the ability for data analyzers to discover and recruit new data sources based on affinities with information neighborhoods. Because this type and classification information is "external", it may be recognized and acted upon by many IoT elements, such as integrator functions and propagator nodes (along with their associated publishing agents, if so-equipped).

In this way, integrator functions monitoring a pressure sensor in a pipeline might seek out nearby temperature sensors to look for correlations that might provide richer information. The type and classification of the chirp packet alone conveys some potential knowledge that may be analyzed and coordinated with other information, and this is carried throughout the network as chirp packet streams are forwarded.

This feature is true even if the transmitting sensors were installed for a different application, by a different organization, or at a different point in time. The option for "public" advertising of type and classification allow broader use (and re-use) of chirp streams, by enabling dynamic publish/subscribe relationships to be created and modified over time as the IoT "learns".

This benefit is achieved without burdening end devices. Because most end devices are by definition very simple in the Internet of Things, those designed to receive IoT chirp packets will be required to process only the most basic of elements of the protocol (for example, using public markers to identify packets addressed to themselves and reading only that data). The IoT elements making much more extensive use of the capabilities of the chirp packet are those that must route or analyze data from many end devices, specifically the propagator nodes and integrator functions. These are the propagator nodes and integrator functions, described briefly next and in more detail in Chapters 4 and 5.

Applying Network Intelligence at Propagator Nodes

As noted previously, replicating even this highly efficient chirp protocol traffic indiscriminately throughout the IoT would clearly choke the network, so intelligence must be applied at levels above the individual end devices. This is the responsibility of *propagator nodes*, which are devices that create an overarching network topology to organize the sea of machine-to-machine interactions that make up the Internet of Things.

Propagator nodes are typically a combination of hardware and software distantly similar to WiFi access points. They handle "local" end devices, meaning that they interact with end devices essentially within the (usually) wireless transmission range of the propagator node. They can be specialized or used to receive chirps from a wide array of end devices. Eventually, there would be tens or perhaps hundreds of thousands of propagator nodes in a city like Las Vegas. Propagator nodes will use their knowledge of adjacencies to form a near-range picture of the network. They will locate in-range nearby propagator nodes, as well as end devices and integrator functions either attached directly to or reached via those propagator nodes. This information is used to create the network topology: eliminating loops and creating alternate paths for survivability.

The propagator nodes will intelligently package and prune the various chirp messages before broadcasting them to adjacent nodes. Examining the public markers, the simple checksum, and the "arrow" of transmission (toward end devices or toward integrator functions), damaged or redundant messages will be discarded. Groups of messages that are all to be propagated via an adjacent node may be bundled into one "meta" message–a small data "stream"–for efficient transmission. Arriving "meta" messages may be unpacked and repacked.

Some classes of propagator nodes will contain a software publishing agent (see Chapter 4). This publishing agent interacts with particular integrator functions to optimize data forwarding on behalf of the integrator. Propagator nodes with publishing agents may be "biased" to forward certain information in particular directions based on routing instructions passed down from the integrator functions interested in communicating with a particular functional, temporal, or geographic "neighborhood" of end devices. (Neighborhoods formed by integrator functions are further described in Chapter 5.) It is the integrator functions that will dictate the overall communications flow based on their needs to get data or set parameters in a neighborhood of IoT end devices.

In terms of discovery of new end devices, propagator nodes and integrator functions will be again similar to traditional networking architectures. When messages from or to new end devices appear, propagator nodes will forward them and add the addresses to their tables (see Figure 2-7). Appropriate age-out algorithms will allow for pruning the tables of adjacencies for devices that go offline or are mobile and are only passing through.

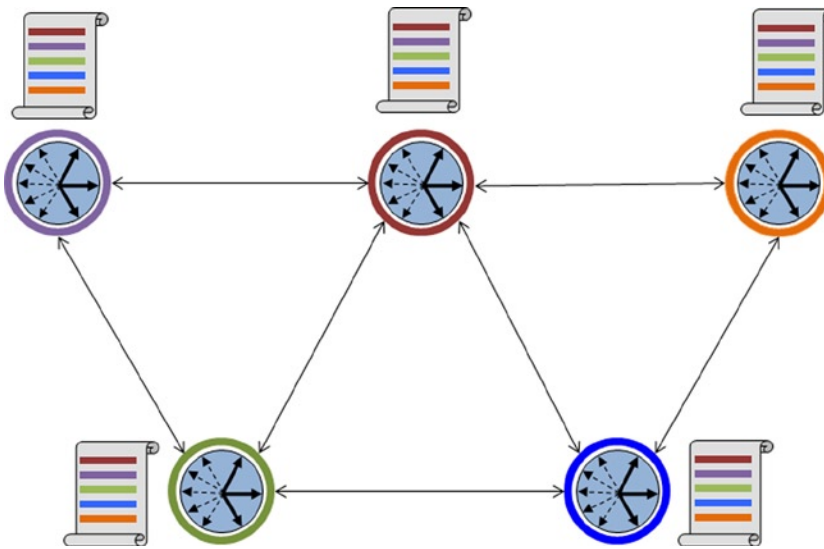

Figure 2-7. *Propagator nodes independently build routing tables (and thus, the network topology) based on the discovery of adjacent propagator nodes. Although not shown here, the location of integrator functions and discovered end devices would also be included in the makeup of the topology*

Transport and Functional Architectures

The emerging architecture of the Internet of Things combines two completely independent network topologies or architectures: *transport* and *functional*, as shown in Figure 2-8. The transport architecture is the infrastructure over which all traffic is moved and is provided primarily by propagator nodes (and the global Internet). The functional architecture is the virtual "zone" or "neighborhood" of interest created by integrator functions independent of physical paths.

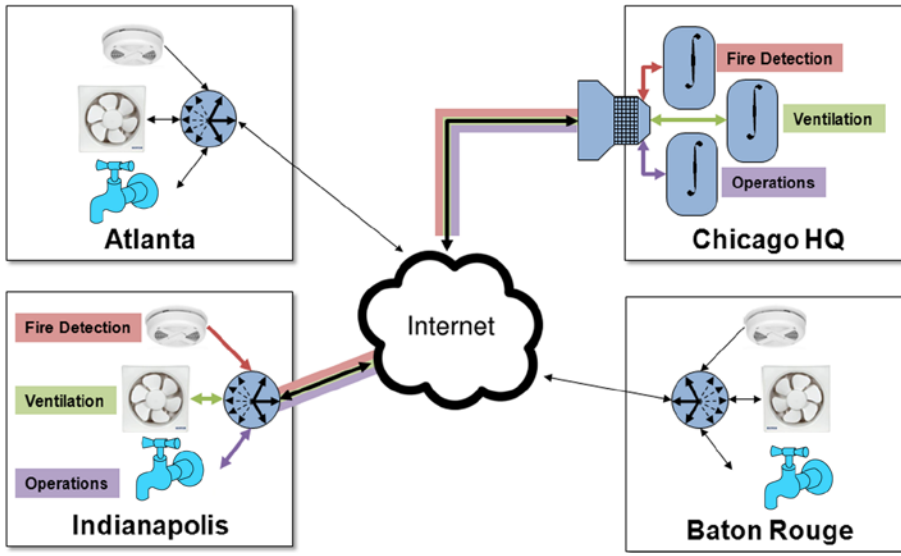

Figure 2-8. *The network topology and logical topology of the Internet of Things can vary considerably*

The transport network portion of the Internet of Things operates with little or no context of the actual significance of the data chirps being handled. As noted previously, propagator nodes build the transport network based on more-traditional networking concepts and routing algorithms (see Chapter 4). End chirp devices may link to propagator nodes in a wide variety of ways: wirelessly via radio or optical wavelengths (see the following "Chirps in a Wireless World" sidebar), power line networking, a direct physical connection, and so on. A single propagator node can be connected to a large number of chirp devices and provide services for all. Unless the propagator node is biased by the integrator function, the basic model is "promiscuous forwarding."

CHIRPS IN A WIRELESS WORLD

One other aspect of communication to be addressed within the Internet of Things is the matter of wireless networking. It's likely that many of the end device chirp connections in the IoT will be wireless, using a wide variety of frequencies and formats. This fact seems to suggest a need for something such as Carrier Sense Multiple Access with Detection (CSMA/CD), as used in 802.11 WiFi. But that's another aspect of traditional networking that must be forgotten.

Again, data rates will be very small, and most individual transmissions are completely uncritical. Even in a location with many devices vying for airtime, the overall duty cycle will be very low. And most messages will be duplicates, from our earlier principle of over-provisioning at the edge through repetition. With that in

mind, an occasional collision is of zero significance. All that must be avoided is a "deadly embrace," in which multiple devices, unaware of one another's presence, continue transmitting at exactly the same time and colliding over and over.

The solution is a simple randomization of transmission times at every device, perhaps with continuously varying pauses between transmissions based on prime numbers, hashed end device address, or some other factors that provide uniquely varying transmission events.

Although the resulting communication scheme is very different from traditional networking protocols, it is all that is necessary for the IoT. Providing just enough communication at very low cost and complexity is a general IoT architectural principle and will be "good enough" for the Internet of Things.

As will be discussed in Chapter 4, propagator nodes bundle and convert chirp traffic as necessary for transport to adjacent propagator nodes and thence to integrator functions or chirp devices. The link between propagator nodes is typically a traditional networking protocol such as TCP/IP, but it can also be chirp-based.

Besides transporting the very simple chirps, the higher-level protocol packets created by the propagator nodes include additional contextual information not found in the chirps. This data may include additional address information related to location, time of day, and other factors, as shown in Figure 2-9. Thus, the propagator nodes increase the utility of the chirp data stream without burdening the vast numbers of end devices with networking cost and complexity. This additional contextual information is added only by propagator nodes and analyzed by integrator functions.

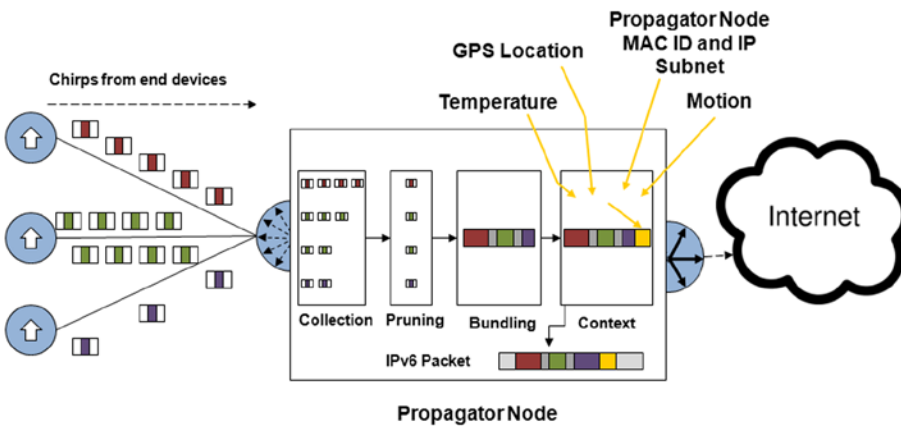

Figure 2-9. *As chirps are bundled within propagator nodes, additional location, addressing, protocol, and other information is added*

An important difference between the IoT transport architecture and many forms of traditional networking is that it is fundamentally egalitarian, similar to wind currents carrying all types of plant pollen. Propagator nodes will forward IoT traffic to and from *any* end device or integrator function within the constraints of "trust," "communications," and "control" factors (these will be outlined in Chapter 6). The IoT can then "piggyback" on existing infrastructure, and each new propagator node may increase functionality for a variety of users and integrator functions. Fundamentally, the transport network topology and architecture does not create (or limit) the *functional* IoT network topology, which is created by integrator functions.

Functional Network Topology

With the transport network architecture (described previously) providing forwarding services for chirps in both directions ("down" toward chirp devices and "up" toward integrator functions), attention may now be turned to the functional IoT architecture, which is overlaid on the transport architecture in somewhat the same way that the propagation of pollen is overlaid on general wind currents in the atmosphere.

The functional network of the IoT, then, becomes less a matter of how the "wires" (physical or virtual) are connected and much more a matter of information that is of interest. The emerging architecture of the Internet is fundamentally a "publish and subscribe" model driven by the integrator functions. It is also receiver-oriented, with the machine at the far end of the transmission "arrow" determining what data is pertinent and useful.

Defined by Integrator Functions

At this point, a brief description of the integrator function is appropriate, with more detail found in Chapter 5. Integrator functions may take a wide array of physical forms, and multiple logical integrator functions can be deployed on one machine with a single connection to the traditional Internet (perhaps via a filter gateway). From a functional standpoint, they are somewhat autonomous creators of relationships with a select group of end points.

As an example, imagine an integrator function designed to monitor moisture content in the far-flung fields of an agribusiness concern (see Figure 2-10). The moisture–sensing end devices broadcast chirps at intervals, indicating the moisture content of the surrounding soil. The tiny chirps of data have a transport "arrow" pointing toward integrator functions.

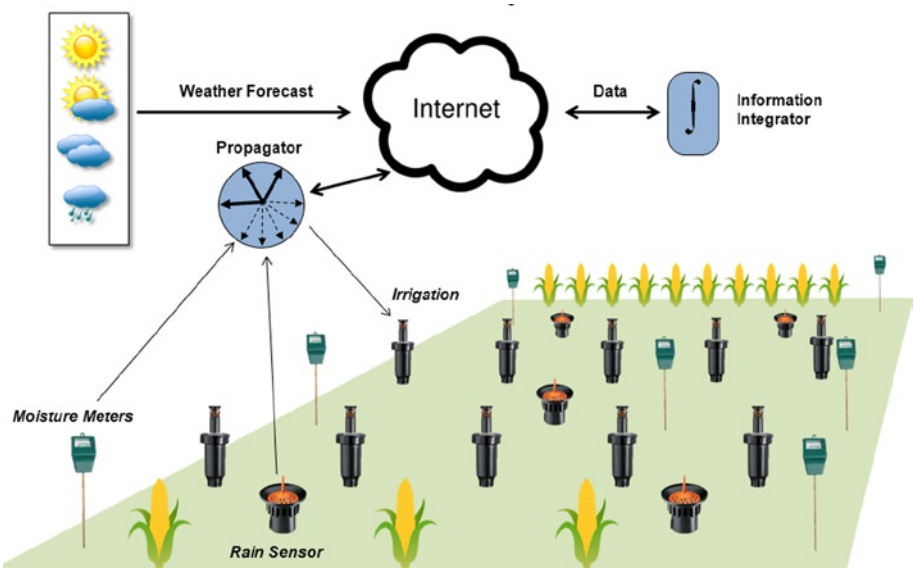

Figure 2-10. *An integrator function retrieves data from end devices such as moisture sensors and external feeds such as the expected precipitation and humidity, using the information to control irrigation valves*

The chirps are received by in-range propagator nodes deployed by the agribusiness concern (or anyone else). As noted previously, these chirps leave the propagator nodes bundled with additional contextual information such as a full IPv6 address and location information, allowing a more precise location and identification of the specific individual sensor that is not available from the simple chirp. The transport network of the propagator node essentially "publishes" these data streams via the traditional Internet.

Harvesting Information from the IoT

The preceding description suggests a virtual private sensor network, with a single agribusiness supplier installing its own end-device sensor propagator nodes, using the traditional Internet to create a routing path, and then monitoring the network privately for its own benefit. And certainly many IoT big data "neighborhoods" are created in this way. But there is also a tremendous potential for building networks that rely on data provided by Internet of Things elements not owned, managed, and controlled by a single source.

In the emerging social networking culture in the Western world, crowd sourcing and data sharing is becoming more commonplace. In light of this, individuals and organizations may choose to install sensors, cameras, and other devices of all kinds locally, providing the IoT streams from these devices generically and publicly. (Note that many individuals and groups do this today with web cams, weather sensors, and the like using traditional Internet protocols such as IP).

Propagator nodes set to promiscuously forward generic chirps would simply move these packets in the general direction of integrator functions. (Note that it is possible for propagator nodes to be used for both private and public streams simultaneously—offering transport for the general good, as it were.)

An integrator function might be configured, then, to gather data from interesting end devices that it has discovered by searching out small data streams from specific classes of device, location, or other characteristics. These integrator functions might combine small data streams from many independent end devices installed by any number of unknown individuals to create interesting new big data information.

Programming and "Bias"

Human programming of the integrator function may instruct it to look for certain locations and types of data streams via the traditional Internet, or the integrator functions may identify potentially interesting candidate data streams through affinities with known sources. Locating appropriate moisture sensor streams on the Internet, the integrator function begins to receive and incorporate this data. The integrator function may even "bias" the publishing agent within propagator nodes (if so-equipped) for some efficiency in combining chirps into larger packets in small data streams or discarding duplicate chirps. (Attached filter gateways might also serve to prune and select from verbose streams in the same way. This topic is more fully discussed in Chapter 5).

The human programming of the integrator function may now incorporate these streams of data on moisture content to look for changes that represent drying out beyond preset thresholds. Additional data, such as weather reports, air temperature, and irrigation reservoir levels (acquired from a variety of sources and feeds, both chirp-based and via the traditional Internet), might also be incorporated to provide a complete picture of irrigation needs for current and future periods of time.

The resulting reports might be provided for human action. Or, in a more automated scenario, the integrator function might respond (via its programming) to change watering times or durations in specific fields (if irrigation valves are also under IoT control). In this application, the integrator function might also analyze video surveillance streams to confirm that sprinklers are on and operating normally.

Note that this functional IoT network might interconnect over any transport topology. The agribusiness need not build out its own private network for the entire transport path; instead, it can use the traditional Internet for much of the transport infrastructure. The enterprise might deploy only the moisture sensors and some specialized propagator nodes, as appropriate.

This is only one example out of millions that might be imagined for the Internet of Things. But the basic principles of very simple devices at the edge, publish-and-subscribe, utilization of public network transport, and integration of a variety of data sources apply broadly.

Receiver-Oriented Selectivity

In the same way that female plants "select" only the appropriate pollen from the same species and reject foreign pollen, dust, and other material, integrator functions are similarly selective in choosing which chirp streams to incorporate as inputs for analysis.

Integrator functions may be programmed to "set," configure, or otherwise manipulate end devices by generating "chirp" traffic of their own that is packaged for routing through the traditional Internet to a propagator known to be near the target end device. With the transmission "arrow" set in the direction of end devices, these packets are transported to the appropriate propagator node (typically within IPv6 packets) and then output as IoT chirps. Integrator functions may combine chirps for widely scattered end devices in a single broadcast packet, which is then pruned and rebroadcast as necessary by intermediate propagator nodes.

The end devices may be able to "hear" a variety of traffic, but thanks to similar receiver-oriented selectivity, they act upon only the specific traffic intended for them. As noted earlier, the intermediate routing and addressing information is primarily a function of the propagator nodes; end devices need only detect the simple IoT chirp addresses.

The following chapter will detail the IoT architecture relating to end devices and will include suggested implementation strategies and alternatives.